UNIVERSIDAD DE ALMERÍA

Computación en procesadores gráficos

José Antonio Martínez García
Francisco M. Vázquez López
Manuel Ujaldón Martínez
Ester Martín Garzón

Objetivos

- Saber explotar los recursos disponibles en los PCs de uso extendido para acelerar algunas aplicaciones:
  - Analizar las claves para explotar eficazmente las arquitecturas de los Procesadores Gráficos desde el punto de vista del programador de aplicaciones de propósito general (GP-GPU computing)
  - Analizar y desarrollar códigos basados en CUDA
- Estudiar las principales tendencias actuales en supercomputación con procesadores gráficos

Contenidos

- Arquitectura de las GPUs
- Modelo de programación SIMT
- Claves computacionales para la programación de GPUs
- Programación basada en CUDA
- Supercomputacion Grafica y Arquitecturas Emergentes

Temporización

Fecha	Sesión	Contenido	Profesor
8/11/2010	18:30-21:00	Presentación / Arquitectura de las GPUs.	EMG/JMG
9/11/2010	18:30-21:00	Arquitectura de las GPUs	EMG/JMG
11/11/2010	18:30-21:00	Modelo de programación SIMT	EMG/JAM
12/11/2010	18:30-21:00	Claves computacionales GPUs. Programación CUDA	EMG/JAM
16/11/2010	18:30-21:00	Claves computacionales GPUs. Programación CUDA	FVL
18/11/2010	18:30-21:00	Programación CUDA	FVL
19/11/2010	18:30-21:00	Programación CUDA	FVL
23/11/2010	18:30-21:00	Programación CUDA	FVL
24/11/2010	18:30-21:00	Supercomputacion Grafica y Arquitecturas Emergentes	MU
25/11/2010	18:30-21:00	Supercomputacion Grafica y Arquitecturas Emergentes	MU
26/11/2010	18:30-21:00	Programación CUDA	FVL
30/11/2010	18:30-21:00	Programación CUDA	FVL

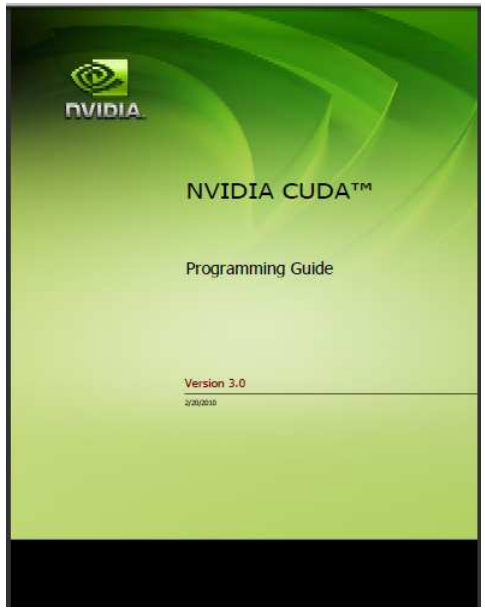
EMG – Ester Martín Garzón
JAM --José Antonio Martínez García
FVL --Francisco Vazquez
MU --Manuel Ujaldon

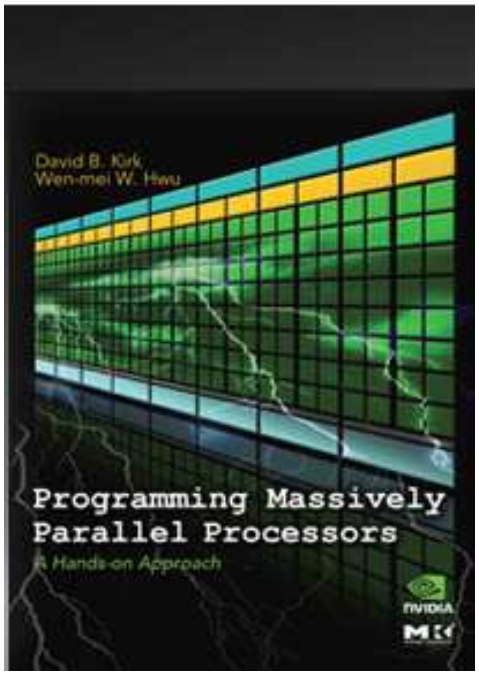
Actividades

- La mayoría de las sesiones se estructuran en dos partes:
  - Exposición de las claves teóricas por parte del profesor.
  - Desarrollo de ejercicios prácticos en el laboratorio

Evaluación y bibliografía

- Evaluación: Basada en el desarrollo de ejercicios prácticos en el laboratorio
- Manual de CUDA y libro





Motivación

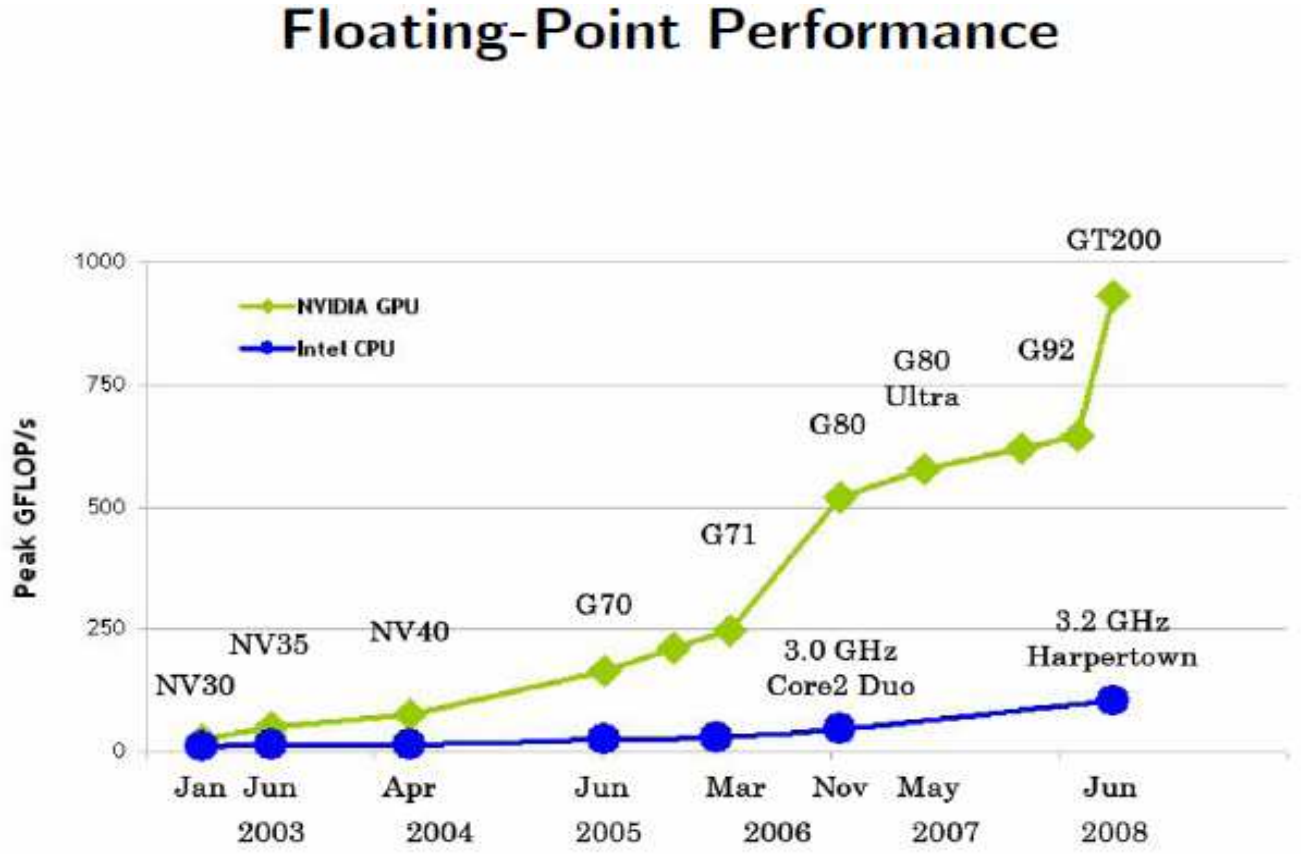


Figure 2: Floating-Point Operations per Second for the CPU and GPU

Motivación

- Se están consiguiendo acelerar multitud de aplicaciones de forma espectacular explotando GPUs de bajo coste

(Cuda Zone)

- Especialmente aplicaciones denominadas "computacionalmente intensivas"➔ Se adaptan al paradigma de programación vectorial (Single Instruction Multiple Data)

Contenidos

- **Arquitectura de las GPUs**
- Modelo de programación
- Claves computacionales para la programación de GPUs
- Programación basada en CUDA
- Supercomputacion Grafica y Arquitecturas Emergentes

Contenidos

- Motivación para usar las GPUs
- Arquitectura de las GPUs:
  - Procesadores
  - Memoria
  - Conexión con la CPU
  - Modelos de GPUs NVIDIA
- Modelo de programación

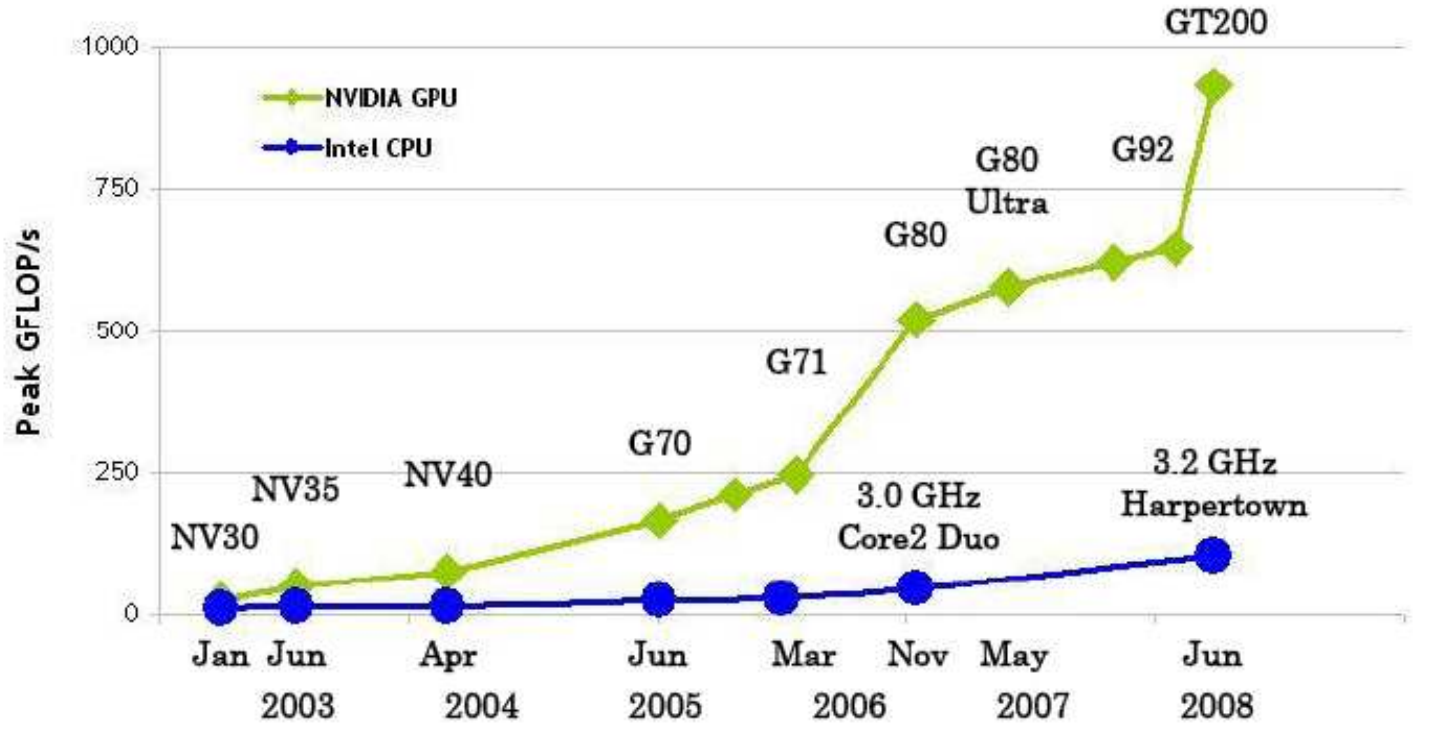
Motivación

(***C**ompute **U**nified **D**evice **A**rchitecture*)

Table 1. NVIDIA GPU technology development.

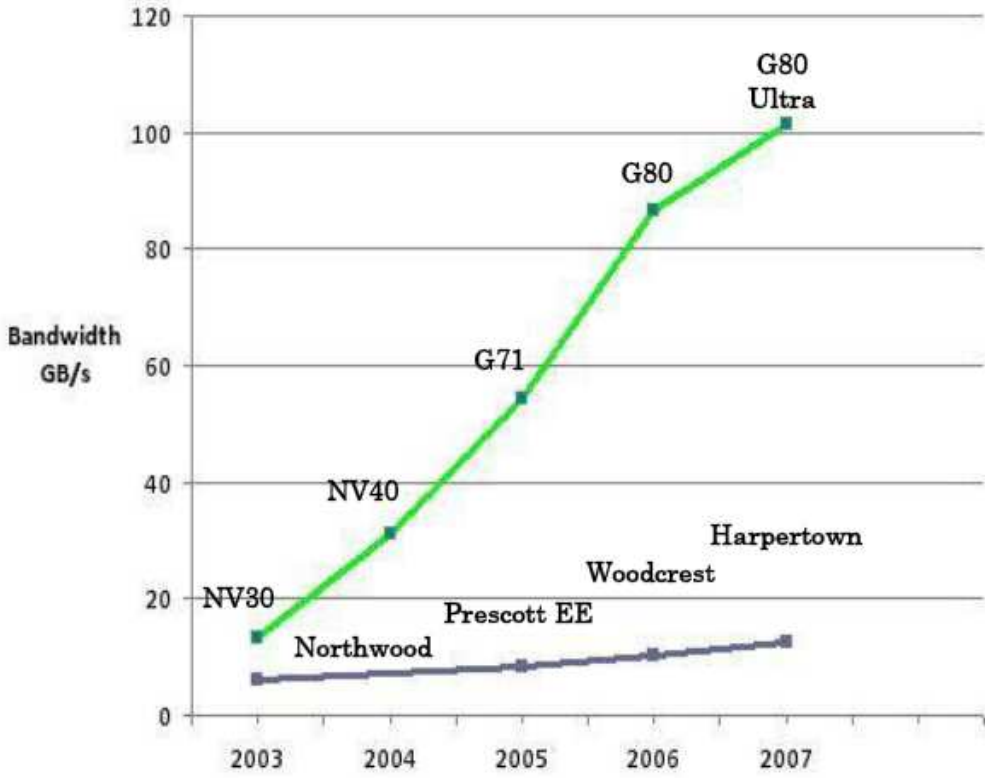
Date	Product	Transistors	CUDA cores	Technology
1997	RIVA 128	3 million	—	3D graphics accelerator
1999	GeForce 256	25 million	—	First GPU, programmed with DX7 and OpenGL
2001	GeForce 3	60 million	—	First programmable shader GPU, programmed with DX8 and OpenGL
2002	GeForce FX	125 million	—	32-bit floating-point (FP) programmable GPU with Cg programs, DX9, and OpenGL
2004	GeForce 6800	222 million	—	32-bit FP programmable scalable GPU, GPGPU Cg programs, DX9, and OpenGL
2006	GeForce 8800	681 million	128	First unified graphics and computing GPU, programmed in C with CUDA
2007	Tesla T8, C870	681 million	128	First GPU computing system programmed in C with CUDA
2008	GeForce GTX 280	1.4 billion	240	Unified graphics and computing GPU, IEEE FP, CUDA C, OpenCL, and DirectCompute
2008	Tesla T10, S1070	1.4 billion	240	GPU computing clusters, 64-bit IEEE FP, 4-Gbyte memory, CUDA C, and OpenCL
2009	Fermi	3.0 billion	512	GPU computing architecture, IEEE 754-2008 FP, 64-bit unified addressing, caching, ECC memory, CUDA C, C++, OpenCL, and DirectCompute

Motivación



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

Motivación



Memory Bandwidth for the CPU and GPU

Motivación

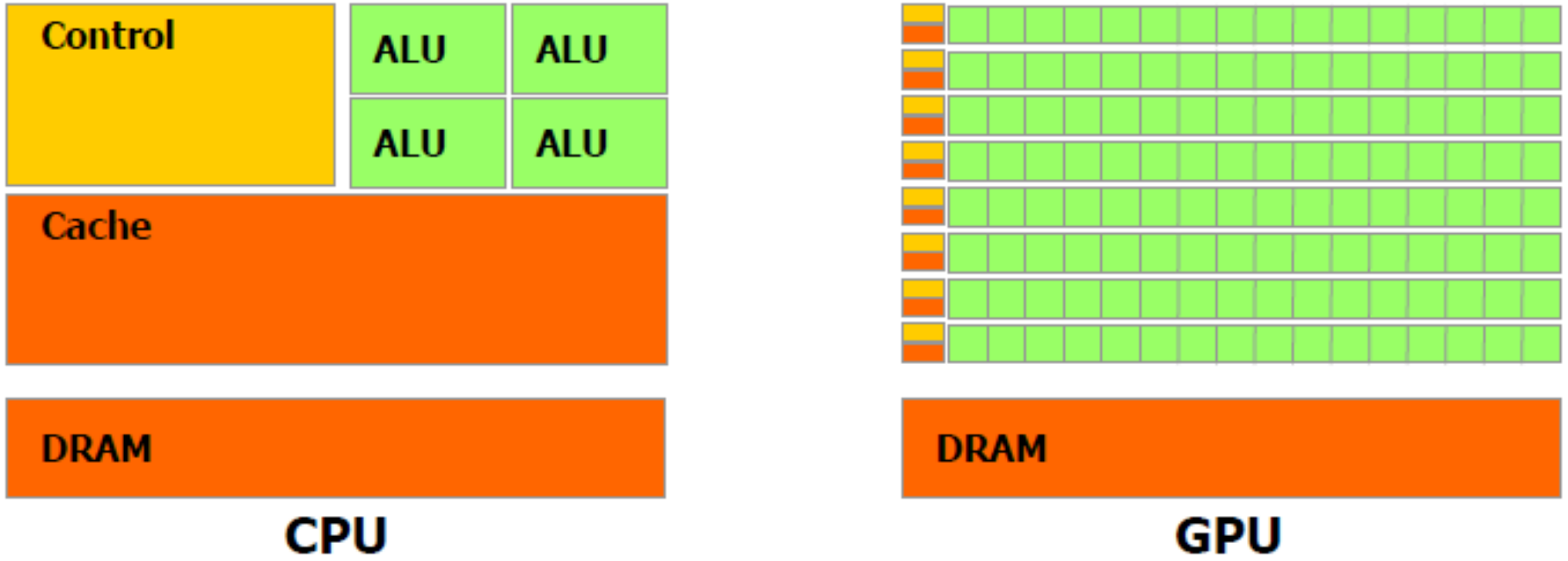


Figure 1-2. The GPU Devotes More Transistors to Data Processing

Motivación

Table 2. CPU+GPU coprocessing execution time, assuming that a CPU core is 5× faster and 50× the area of a GPU core.

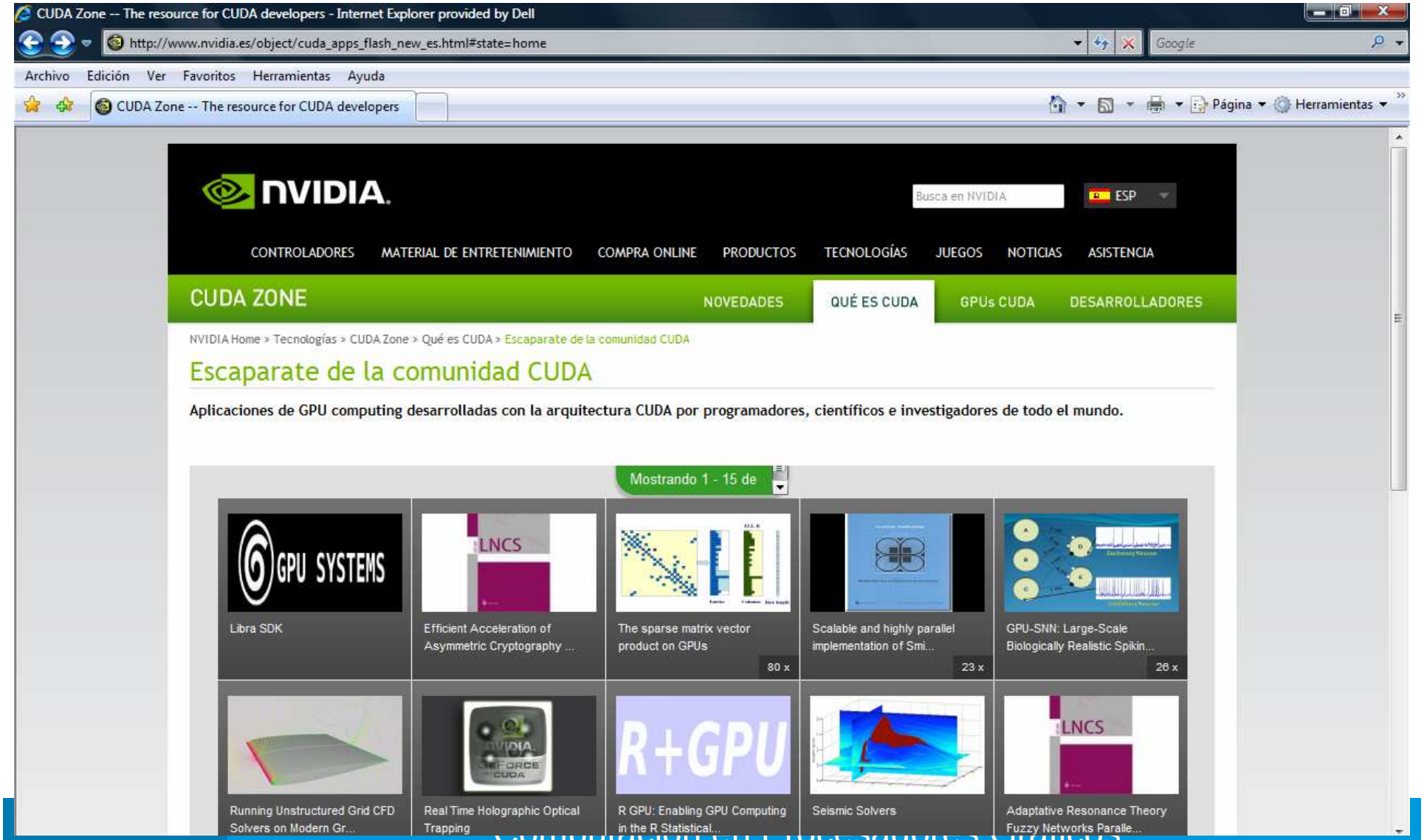
	Configuration	Processing time for 1 CPU core	Processing time for 500 GPU cores	Processing time for 10 CPU cores	Processing time for 1 CPU core + 450 GPU cores
Program type	**Area**	**50**	**500**	**500**	**500**
Parallel-intensive program	0.5% serial code	1.0	5.0	1.0	1.00
	99.5% parallelizable code	199.0	0.4	19.9	0.44
	Total	200.0	5.4	20.9	1.44
Mostly sequential program	75% serial code	150.0	750.0	150.0	150.0
	25% parallelizable code	50.0	0.1	5.0	0.11
	Total	200.0	750.1	155.0	150.11

Motivación

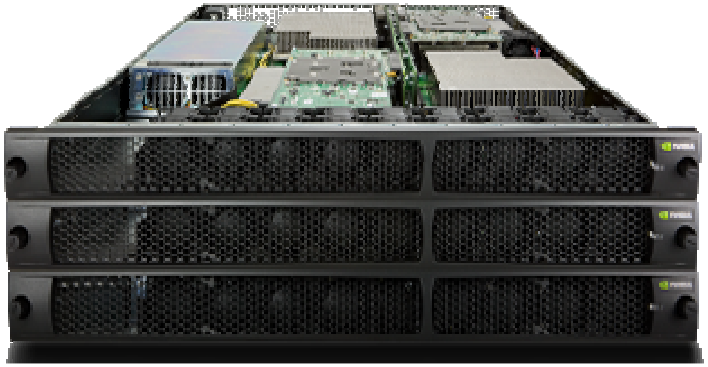
Table 3. Representative CUDA application coprocessing speedups.

Application	Field	Speedup
Two-electron repulsion integral[12]	Quantum chemistry	130×
Gromacs[13]	Molecular dynamics	137×
Lattice Boltzmann[14]	3D computational fluid dynamics (CFD)	100×
Euler solver[15]	3D CFD	16×
Lattice quantum chromodynamics[16]	Quantum physics	10×
Multigrid finite element method and partial differential equation solver[17]	Finite element analysis	27×
N-body physics[18]	Astrophysics	100×
Protein multiple sequence alignment[19]	Bioinformatics	36×
Image contour detection[20]	Computer vision	130×
Portable media converter*	Consumer video	20×
Large vocabulary speech recognition[21]	Human interaction	9×
Iterative image reconstruction[22]	Computed tomography	130×
Matlab accelerator**	Computational modeling	100×

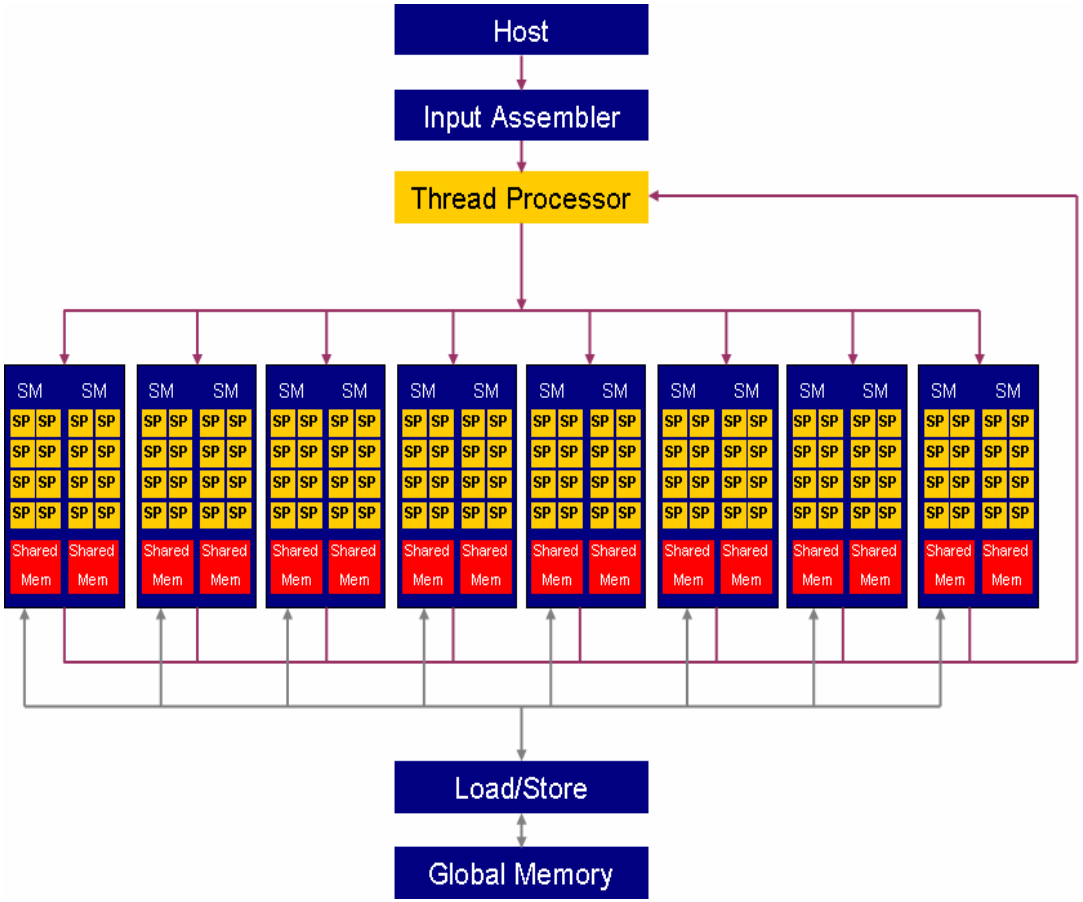
* Elemental Technologies, Badaboom media converter, 2009; http://badaboomit.com.
** Accelereyes, Jacket GPU engine for Matlab, 2009; http://www.accelereyes.com.


CUDA Zone -- The resource for CUDA developers - Internet Explorer provided by Dell
http://www.nvidia.es/object/cuda_apps_flash_new_es.html#state=home
Archivo Edición Ver Favoritos Herramientas Ayuda
CUDA Zone -- The resource for CUDA developers
NVIDIA
Busca en NVIDIA
ESP
CONTROLADORES MATERIAL DE ENTRETENIMIENTO COMPRA ONLINE PRODUCTOS TECNOLOGÍAS JUEGOS NOTICIAS ASISTENCIA
CUDA ZONE
NOVEDADES QUÉ ES CUDA GPUs CUDA DESARROLLADORES
NVIDIA Home > Tecnologías > CUDA Zone > Qué es CUDA > Escaparate de la comunidad CUDA
Escaparate de la comunidad CUDA
Aplicaciones de GPU computing desarrolladas con la arquitectura CUDA por programadores, científicos e investigadores de todo el mundo.
Mostrando 1 - 15 de
GPU SYSTEMS
Libra SDK
LNCS
Efficient Acceleration of Asymmetric Cryptography ...
The sparse matrix vector product on GPUs
80 x
Scalable and highly parallel implementation of Smi...
23 x
GPU-SNN: Large-Scale Biologically Realistic Spikin...
28 x
Running Unstructured Grid CFD Solvers on Modern Gr...
Real Time Holographic Optical Trapping
R+GPU
R GPU: Enabling GPU Computing in the R Statistical...
Seismic Solvers
LNCS
Adaptative Resonance Theory Fuzzy Networks Paralle...

Arquitectura de las GPUs



Arquitectura de las GPUs



- **Thread Processor**
- **Streaming Multiprocessor: SMs**
- **Scalar Processor: SPs**

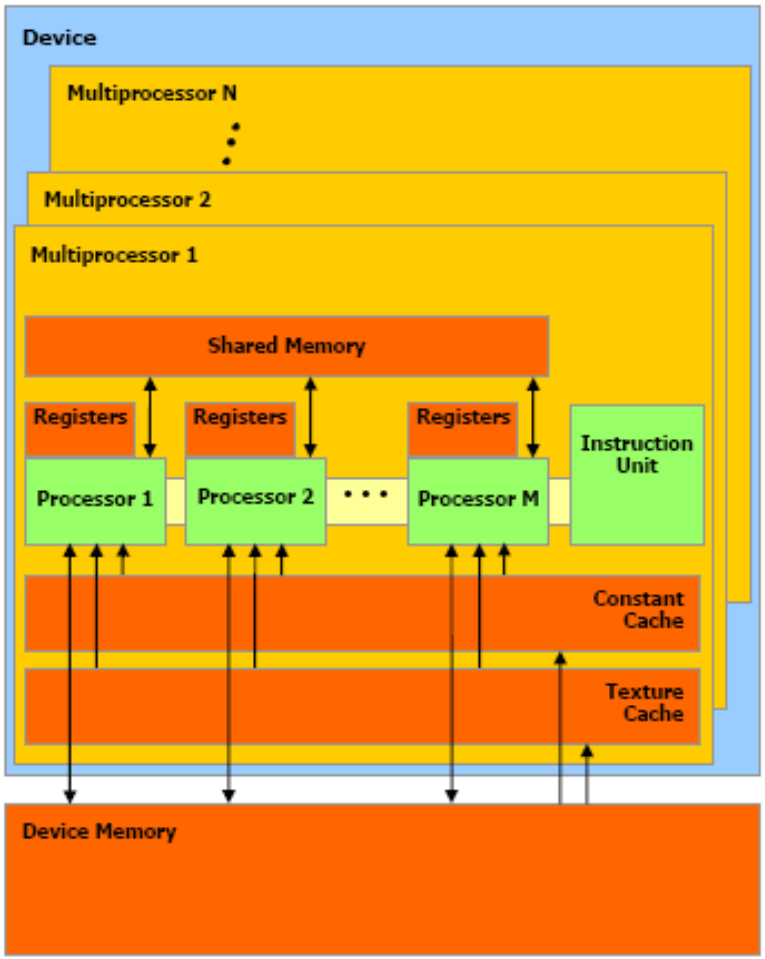
Arquitectura de las GPUs

- **Thread Processor**: Coordina la ejecución de los threads
- **Streaming Multiprocessor SM**:
  - Contiene 8 Scalar Processor SP
  - 4, 14, 16, 30 SMs según arquitectura (G80, G92,…., GT200)
  - GeForce GTX 285: 30 MP x 8 SP = 240 Cores
  - GeForce GTX 295: Dual 30 MP x 8 SP x 2 GPUs = 480 Cores
  - Tesla C1070: 30 MP x 8 SP x 4 GPUs = 960 Cores
  - Fermi (GT300): 16 SMs x 32 SPs = 512 Cores

Arquitectura de las GPUs

- **Scalar Processor SP**: Cores
  - Unidades funcionales en coma flotante de simple precisión
  - 2 Unidades en coma flotante de doble precisión IEEE 754-1985
  - Fermi: 4 Unidades conforme estándar IEEE 754-2008

Arquitectura de las GPUs



A set of SIMT multiprocessors with on-chip shared memory.

- **Memoria**
  - **On-chip:**
    - **Registros**
    - **Shared Memory**
  - **Off-chip:**
    - **Memoria global o device memory**
    - **Texture cache, constant cache**

Arquitectura de las GPUs

- **Memorias On-Chip**: Baja latencia (varios ciclos)
  - Registros (R/W): Almacenan las variables declaradas en los kernels
    - Accesible por thread
    - Tamaño: 16 K Registros de 32 bits
  - Shared Memory (R/W): Almacenan arrays
    - A compartir entre todos los threads del bloque
    - Comunicación entre todos los threads del bloque
    - Tamaño: 16 Kb (Fermi: 48Kb + 16Kb configurable)

Arquitectura de las GPUs

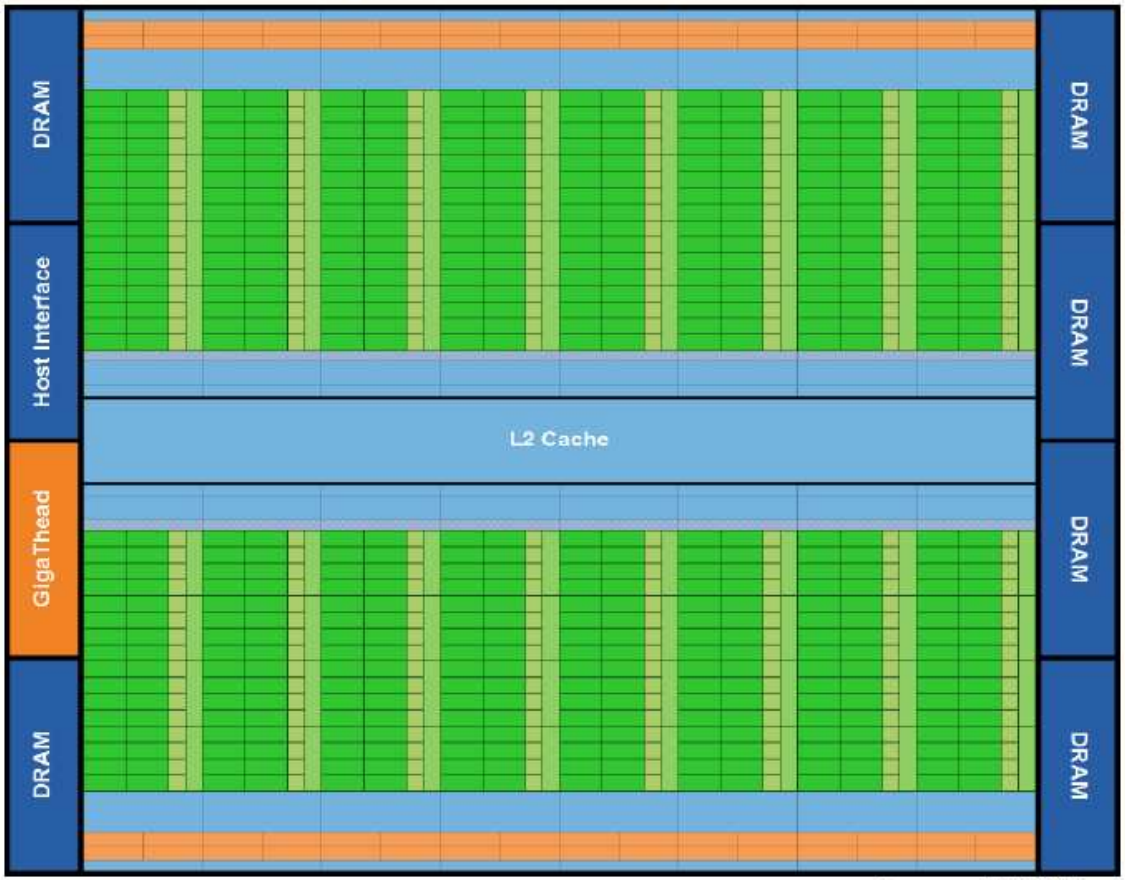
- **Memorias Off-Chip**:
  - Device memory (R/W)
    - Tamaños: 512Mb, 1GB, 2GB, 4GB
    - Alta latencia: 600-700 ciclos por acceso aleatorio
    - Mejorar acceso bajo condiciones de coalescencia
  - Caché de texturas (R) 6Kb-8Kb por MP
    - Zona ligada a device memory pero 8Kb dentro de cada MP
    - Misma latencia que la de un registro en cache hit
    - Misma latencia que device memory en cache miss
    - Reserva dinámica
  - Caché de constantes (R) 64Kb
    - Mínima latencia cuando todos los threads acceden a la misma dirección
    - Tamaño fijo. Reserva estática

Arquitectura de las GPUs

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double Precision Floating Point Capability	None	30 FMA ops / clock	256 FMA ops /clock
Single Precision Floating Point Capability	128 MAD ops/clock	240 MAD ops / clock	512 FMA ops /clock
Special Function Units (SFUs) / SM	2	2	4
Warp schedulers (per SM)	1	1	2
Shared Memory (per SM)	16 KB	16 KB	Configurable 48 KB or 16 KB
L1 Cache (per SM)	None	None	Configurable 16 KB or 48 KB
L2 Cache	None	None	768 KB
ECC Memory Support	No	No	Yes
Concurrent Kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit

Arquitectura de las GPUs

Figure 1. Fermi's Major Elements

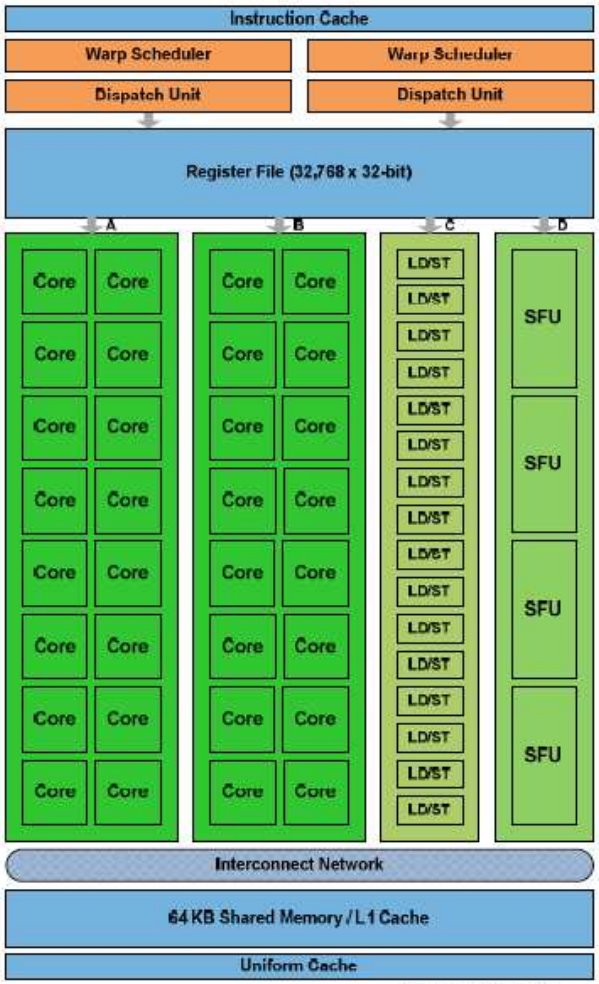


Source: NVIDIA

Arquitectura de las GPUs

Streaming Multiprocessors

Figure 2. Fermi SM



Source: NVIDIA

- Arquitectura Fermi

- http://www.nvidia.com/object/fermi_architecture.html

ACTIVIDADES. 1

- Componentes CUDA:
  - Basicos: Driver + Toolkit
  - Opcionales: SDK
  - http://developer.nvidia.com/object/cuda_3_0_downloads.html
- Driver: Propietario de NVDIA pero de libre distribución.
- Toolkit: Basado en gcc:
  - Nuevas declaraciones.
  - Nuevas funciones.
  - Nuevas tipos de datos.
- Extensiones de declaracion de funciones en CUDA
  - `__device__ float DeviceFunc()Exe D Call D`
  - `__global__ void kernelFunc() Exe D Call H`
  - `__host__ float HostFunc() Exe H Call H`

- Actividad: Generación de la librería de utilidades del SDK.Visualizar y estudiar algunos ejemplos.
- Actividad: Template de un programa CUDA: curso01.tgz
  - Ficheros .cu
  - Ficheros .h
  - Ficheros .cpp
  - Makefile
  - http://dali.ace.ual.es/~jmartine/curso01.tgz
- Actividad: Modificar el archivo makefile para adaptarlo a nuestro entorno.
- Actividad: Identificar que funciones son ejecutables por el host y cual por la tarjeta NVIDIA. Modificar las declaraciones de las funciones acorde a lo establecido en el toolkit de CUDA.

UNIVERSIDAD DE ALMERÍA